

System and Method for Interruption-Free File Access

Field of the Invention

This invention relates in general to a system and method for implementing file access service free from interruption and, in particular, to a system and method for providing such service to a large number of clients simultaneously seeking information service via a wide area network.

Background of the Invention

As a result of increased popularity of the Internet, network services offered by the industry aimed at providing simultaneous service to a vast number of users become ever popular. Large number of users scattered all around who are subscribed and connected to the server system of the same information service provider via a wide area network (WAN) present a unique phenomenon of network activity as they access the service simultaneously. The network activities concentrated on the server system of the service provider are characterized by the vastly diversified file access operations directed to various files in the system storage. Such accesses include both the read-out and write-in of information files originated by the different requirements of each of the accessing users.

Server systems installed in many information service organizations are required to handle heavy, sometimes extremely heavy, access loads originated by the clients they serve. Internet-based stock brokerage service provided by security houses, Internet access service by Web portals, various services by different ISP (Internet Service Provider), and on-line gaming hosted by on-line game service companies are typical examples of network services that provide simultaneous access service to hundreds of thousands of users. At a popular ISP, for example, up to several million accesses per second is possible. To satisfy these high access rates, adequate file access management system is necessary. Without managed high-efficiency access service, either customers are likely to be lost due to slow service or huge investment are required to keep up with the growing service demand. Lost customers represent lost business while investments on inefficient hardware and/or software eat away profits. Therefore, the removal of

access bottlenecks in the network information service industry becomes one of the most important issues for healthy and prosperous business operation.

For instance, at a major news event, vast number of viewers rush to news websites to access the news details. In these accesses to a news website, it can be expected that news information downloads greatly out-number the upload of readers' opinions. As another example, consider the situation in which crowds of players log into an on-line game hosting company to participate in the same game in non-office hours. Relatively heavy upload accesses to the game server system as compared to the game information download accesses can be expected as the game unfolds for each of the individual players. In the case of news access, more people read news than those post comments. However, in the case of on-line game, the upload accesses for game information in relation to each of the participating player are much more intense than uploading of reader opinion in news websites. This is because a fascinating on-line game involving the participation of tens of thousands of players has to reflect the particulars of each individual player concerning his or her action, location and other parameters. In general, complexity of a game can be translated into attractiveness to players since dull games are not likely to be fascinating.

Conventionally, efficiency of file read accesses to disk storage in a computer system can be substantially improved utilizing techniques including disk caching and proxy service. However, information writing to the disk storage system, in general, do not enjoy these or similar techniques. Write caching is possible but at substantial risks. For maximum data integrity, all data are written into the storage subsystem immediately. Hardware writing operation for a specific piece of information into a disk-based storage system takes substantially longer time than reading it. This has been an inherent characteristic for modern magnetic disk-based storage equipment. As a result, the more frequent the data writing operation is in a database server, the larger bandwidth is required against the data bus channel where the storage system resides in the computer system. This is particularly the case for applications such as on-line game hosting. As mentioned, the more fascinating the game is, the more write accesses are expected in the system.

Thus, pace of the main program running on a server system frequently becomes

sluggish when there are a vast number of users requesting information access simultaneously. On some occasions, execution of the main program even becomes completely stalled. Modern computer systems devote relatively less time in the computation-related processing as the central processing units become more powerful and operate at increased clock rates. In contrast, time required for the information I/O in the disk-based storage subsystems becomes the bottleneck of the entire operation. A disk-intensive application is certain to take more time to complete than one that requires less disk accesses. Mechanical head movements in a disk drive take significantly more time than the CPU performing arithmetic and logic operations.

Meanwhile, although magnetic disk-based storage devices have been advancing in terms of storage density and thus increased storage capacity, however, average access times, both read and write, in these devices have not been improved in parallel. Limitation to the speed of mechanical movements becomes the constraining barrier for disk performance improvements. Thus, the issue of how the penalty to system overall operation speed caused by frequent disk accesses can be reduced has become one of the primary considerations for the improvement of overall system performance.

For example, consider again the situation in which thousands of players participate simultaneously in an on-going on-line game episode. As the story of the game unfolds, each player submits his or her respective status parameters and retrieve his or her required information based on what the player's role and where the character is situated in the virtual game environment. All these parameter submission and information retrieval are implemented via disk write and read accesses respectively in the game-hosting server system. If, for example, due to the overwhelming number of players logged in and participating simultaneously in the game as frequently is the case, the pace of game story development becomes sluggish or even stalled occasionally as a result of the intensive disk accesses taking place in the system. Quality of the game suffers as a result and less fun of game participation can be appreciated. Therefore, the question of how these sluggishness caused by the inevitable heavy disk accesses can be reduced or even removed is one of the most important service quality factors for an on-line game hosting company.

For applications such as database information access, screen display stalls do not

necessarily imply inconvenience for retrieval of the desired information. However, reduction of the occurrences of display sluggishness and/or stall in display screen, especially graphical user interface (GUI) screens, is beneficial. Removal of display sluggishness and/or stall implies improved overall system performance, which is welcome by both the user and the information service provider.

Figure 1 is a simplified block diagram schematically showing how a main program 120 of an information service system 100 according to prior art interacts with its storage subsystem 140. In the conventional software system, the main program 120 implements direct interface with the storage subsystem 140 of the system hardware whenever an information I/O is required. Such an approach is direct and simple, but waiting periods are inevitable for I/O accesses involving substantial amounts of information. Length of the waiting period grows proportionally with the amount of information accessed. If the waiting amounts up to a length noticeable to the user of the program, smoothness of program pace becomes jeopardized. For a user, it appears to be a program interruption. During the waiting period, however, the CPU of the system is frequently left unoccupied except to idle and wait for the I/O access to complete.

Figure 3 is a flowchart illustrating the process flow implementing a prior-art method for providing simultaneous services to multiple users. If such a scheme is employed to service hundreds or even thousands of simultaneous users, program execution stalls are inevitable and frequently unacceptable for users. This is because each implemented I/O access (positive result of the test step 330) as a result of the request of any of the users will incur a waiting phase (step 340). Program execution halts for an I/O access until the accessed result is obtained. When the number of I/O access requests are excessive, the service quality becomes degraded seriously. Costly measures of purely promoting storage equipment performance does not resolve the problem.

One of the few applicable approaches conventionally known for tackling this problem is to throw in high-level server systems at premium costs to improve hardware performance. Powerful but expensive server systems can certainly perform their assigned tasks faster than cheaper ones, however, hardware capabilities are limited. Raw speed can still be overwhelmed by large number of simultaneous accesses pouring

in from all over the network. Therefore, it is desirable to have a simple and easy method for removing information access bottlenecks in network-based information service systems that can be implemented utilizing inexpensive hardware.

Summary of the Invention

The present invention provides an information service system for providing interruption-free information access service for servicing multiple users communicating to the system via a communications network. The system comprises a main program subsystem, an information access agent subsystem and an information storage subsystem. The main program subsystem executes a main program for providing information access service to the users by receiving user-issued I/O access requests. The information storage subsystem stores information required for implementing the information service. The I/O access agent subsystem processes user-issued I/O access requests by registering the requests received by the main program subsystem and submits the received requests to the information storage subsystem for implementing the read or write accesses corresponding to the requests. The I/O access agent subsystem relays the result of the requests to the main program subsystem for returning to the users upon completion of the submitted requests. The system continues services to user even when I/O access services are pending their corresponding results.

The present invention further provides a method for providing interruption-free file access service to users communicating to an information service via a communications network in an information service system comprising a main program subsystem, an information storage subsystem and an I/O access agent subsystem. The method comprises the steps of: a) the main program subsystem registering information access requests issued by users in a list and submitting the requests to the I/O access agent subsystem; b) the I/O access agent subsystem submits the requests issued by users to the information subsystem for implementing read or write accesses corresponding to the requests; and c) the I/O access agent subsystem relays the result of the requests to the main program subsystem for returning to the users upon completion of access of the submitted requests by the information storage subsystem, wherein the information

service system continues information service for users even when an I/O access is pending.

Brief Description of the Drawings

Other objects, features and advantages of the present invention will become apparent by way of the following detailed description of the preferred but non-limiting embodiments. The description is made with reference to the accompanied drawings in which:

Figure 1 is a simplified block diagram schematically showing how a main program in a prior-art system interacts with its storage subsystem;

Figure 2 is another simplified block diagram schematically showing how the main program in a service system in accordance with the teaching of the present invention interacts indirectly with its storage subsystem for information access;

Figure 3 is a flowchart illustrating the process flow implementing the conventional method for providing simultaneous services to multiple users;

Figure 4 is a flowchart illustrating the process flow implementing the method in accordance with a preferred embodiment of the present invention for providing simultaneous I/O access services to multiple users;

Figure 5 is a flowchart illustrating the process flow implemented by an I/O access agent that can be incorporated in the process flow of Figure 4 in accordance with a preferred embodiment of the present invention;

Figure 6 is a simplified block diagram schematically illustrating the system configuration of an I/O access agent subsystem in accordance with a preferred embodiment of the present invention for providing expanded service capability; and

Figure 7 is a simplified block diagram schematically illustrating the system configuration of another I/O access agent subsystem in accordance with an embodiment of the present invention for providing game information service in an on-line game service system.

Detailed Description of the Invention

In the core processing section of modern computer systems, the advancements in

microprocessor technology has been tremendous. Compared to the 16-bit microprocessor operating at 4.77 MHz clock rate in IBM's first personal computers, the microprocessor has evolved into ones with 32 even 64-bit data-path widths operating at clock rates faster than 1GHz and processing instructions in multiple pipelines. The performance has improved more than two, even three, orders of magnitude. However, during the same period of time, the magnetic disk drive technology, although having comparable improvements in the disk storage density as well as overall storage capacity, has not been improved comparably in data access time due to inherent mechanical limitations in the disk read-write head mechanism. Frequently, such mechanical limitations in disk storage data access time comprise bottlenecks in the processing of information in computer systems. This is particularly true in applications involving the processing of multimedia information which requires vast amounts of audio and/or video data.

The method and system of the present invention fully exploit the processing power of modern microprocessors in order to improve overall system performance of an information service system provided over a WAN such as Internet. The method and system of the present invention achieve this by effectively discarding or, in other words, making use the waiting periods associated with the file accesses to the service information storage subsystem initiated by the users from over the WAN. A simplified block diagram for such a system implementing the method of the present invention is depicted in Figure 2.

As is illustrated, the information service system 200 is set up to provide information service for multiple users 271, 272, ... and 279 connected over the WAN 260 via a communication channel 250, preferably a broad-band channel. In one specific application, the information service system 200 can be used as the server system for an on-line game company. In such an application, game players 271, 272, ... and 279 scattered remotely over the WAN 260 can connect to the system 200 in order to participate interactively in a complex and fascinating game.

The information service system 200 suitable for such applications may comprise a user interface 210, a main program 220, a service information storage subsystem 240, and an I/O access agent 230, which stands between the main program 220 and the

storage subsystem 240. Note, as is comprehensible for those in the art, that more functional blocks and/or modules such as that responsible for controlling log-in and service fee charging functionality against a large number of simultaneous users are not shown in the block diagram. Since such functionality are not directly related to the disclosure of the present invention, they can be considered, for example, to be grossly included in the main program 220 of the system.

Also note that the constituent component blocks 210, 220, 230 and 240 in the system 200 of Figure 2 may be conceptually envisioned as blocks of software modules or hardware logic circuit modules. Both are suitably applicable for the description of the method and system of the present invention in the broad sense.

Unlike in the prior-art system of Figure 1, in which the main program 120 issues information access requests directly to the service information storage subsystem 140, the information service system 200 of the present invention as depicted in Figure 2 employs a different and indirect approach. As any one or more of the remote users 271, 272 ... and 279 issues the information access request for either reading or writing a large amount of information out of or into the storage subsystem via the user interface 210, the main program 220 issues the corresponding access request to the storage subsystem 240 indirectly through the I/O access agent 230.

When the I/O access agent 230 acknowledges its acceptance of an information access request in the storage subsystem 240 issued by the main program 220, the agent 230, instead of requesting the storage subsystem 240 to perform an immediate access and waits for the completion of the I/O access, records the request in a cycling list in the first place. In a typical example, all the information access requests as issued by the main program 220 are entered into a plain cycling list in an order determined by the order of occurrence of each of the requests. A cyclically-repeating routine performed by the I/O access agent 230 that scans through all the entries in the record list submits each of the requests to the service information storage subsystem 240 for actually accessing the information.

After the I/O access agent 230 accepts and records an information request that asks for specific information in the storage subsystem 240, but before the agent 230 actually receives the request result from the storage subsystem 240, the main program

220 is allowed to proceed with its processing of providing service for fulfilling the other requests of other users logged into the system 200. In other words, it is absolutely unnecessary for the main program 220 to suspend all its activities simply to wait for the conclusion of an on-going information access in the storage subsystem 240.

5 As the I/O access agent 230 cycles back to the same request in the cycling list, a status flag signifying the conclusion of information access in the storage subsystem 240 is checked for its flag status. If the flag status indicates that the sometimes-lengthy information access has not yet been concluded, the cycling routine proceeds to the next request instead of waiting for its conclusion. This cycling routine is repeated over and over again and, in the process, each of the requests awaits its conclusion. Once a
10 specific request for information access is fulfilled as its flag status is checked to be set, its entry in the cycling list can then be removed.

As the name implies, the I/O access agent 230 functions as an agent for receiving and managing I/O requests issued by various users from over the WAN 260. The agent
15 230 passes the requests on to the storage subsystem 240 of the service system 200 in a managed manner. When processing of a requested I/O is concluded by the storage subsystem 240, the agent 230 then returns the result of the requested I/O back to the specific requesting user, i.e. one of users 271, 272, ... and 279. The I/O access agent 230 performs its tasks of managing the reception of incoming I/O requests and issuing of
20 outgoing I/O results in a continuous program loop. In the process, each I/O access request received by the agent 230 is passed on to the storage subsystem 240 and then processing of the subsequent request continues without holding on the program execution to wait for the current access results.

In such a scheme, each requesting user gets rapid program attention as hosted by
25 the main program 220 even though the access results of any other users are still on the way and are not yet available. I/O access agent 230 does not pause and wait for any specific I/O request results to come up from the storage subsystem 240 and stalls the entire request reception routine.

Figure 4 is a flowchart illustrating the process flow implementing the
30 above-described method in accordance with a preferred embodiment of the present invention for providing simultaneous I/O access services to multiple users. As

described above, the main program (220 in Figure 2) of an information service system (200) operates a repeating program loop for providing information access service to a number of users (271, 272, ... and 279) connected across a WAN (260). In an embodiment as outlined in the program flowchart illustrated in Figure 4, the program loop executed by the main program (220) of the service system (200) initiates for a first I/O access-requesting user at step 410. It should be noted that the program loop can be set up to cycle through all the received users requesting for I/O access service as recorded by the I/O access agent (230).

In general, in a most straight-forward approach, the looping can be set up in a scheme that is based on the simple principle of first-come first-serve. In an embodiment, each new user requesting for I/O access service can be added to a list maintained by the I/O access agent (230). All users in the service list are then scanned through in subsequent program cycles. However, as is comprehensible for those skilled in the art, weighted service loops are also possible. For example, those I/O requests posting longer time in the service loop may be allowed promoted priority to enjoy relatively more frequent status checks than those newly-posted ones. Or, those I/O access service requests asking for certain pre-categorized critical information may be allowed to enjoy higher frequency of status checks than others. This is to ensure that certain application-critical I/O accesses get more attention than those that are not as critical and therefore have a better opportunity of being completed as soon as possible.

After the program is initiated at step 410 for the processing of the request brought up by the first user in the list, program flow then proceeds to step 412 to check if the request has been registered as one for an I/O access. A negative test result of step 412 signifies the fact that the process request of the first user is non-I/O. The program flow may thus be transferred to step 420 for actually and directly performing this non-I/O request for the user. If, on the other hand, the test result of step 412 shows that the user's request is indeed one for information I/O, the program can then be advanced to step 414 to further check if the I/O requests has been submitted for processing and yielded a result from the information storage subsystem. If the accessed result has become ready, program flow may then be advanced to step 416 to update the I/O result and further proceeds to step 420 so that the main program may substantiate the user's

request by issuing the accessed I/O result back to the user.

At this moment when the processing of step 420 is concluded, the program flow can be advanced to step 430, where the necessity of actually submitting for an I/O access is checked. If step 430 determines that there is no such need, the process flow that handles the current user's request can be concluded and program flow passed on to step 450. If, however, step 430 decides that the current user request is indeed a pending I/O access, the program flow can then be advanced to step 440, where an I/O access as implemented by the service information storage subsystem of the system of the present invention can be engaged via the I/O access agent. When this is concluded and the targeted I/O result becomes ready, the program flow is then ready to be concluded for the current user being processed. The program flow then proceeds to step 450, whose operation is described in detail in the following paragraphs. Note, however, that the processing details in step 440 are explained with reference to Figure 5 of the drawing.

Meanwhile, if the test step 414 determines that the user's I/O request has not yet yielded its result, program flow may then be transferred to step 450, where the main program of the system checks to see if another user has come up to request for service. If there is a second user already waiting in the list and requesting for another service to be processed by the system, the program flow is transferred back to step 412 to see if this second user's request is I/O-related, and the program flow continues in the process as described above. If, on the other hand, no subsequent user with his or her request is waiting to be processed at this moment, the main program is transferred back to step 410, where the processing restarts from the first user in the list.

Thus, if the program flow at step 450 determines that there are second, third and further more users waiting in the list to be processed, the program flow may be restarted from step 412 for each of the waiting users in order to fulfill their respective requested I/O processing. If the last user in the list has been reached, the program flow may be returned back to process the first user in the list, and the flow of Figure 4 can be repeated in another cycle.

Here, notice that if a user's requested I/O processing has not been concluded when tested in step 414, program flow of the main program module of the system is just skipped to the next user in its program loop instead of being halted to wait for the

requested result to come up. Thus, the processing power of the system is not wasted in idling processor cycles waiting for the slow I/O to conclude. Instead, the processing power of the system can be directed to other tasks waiting to be processed.

Figure 5 is a flowchart illustrating the process flow implemented by the I/O access agent (230 of Figure 2) that can be incorporated in the process flow of Figure 4 in accordance with a preferred embodiment of the present invention. In Figure 4, when the program processing flow is transferred to step 440 based on the positive test result of step 430, the I/O access agent comes into play so as to substantiate service to the I/O access request for the requesting user.

In order to substantiate the I/O access, the program flow starts at step 510 of Figure 5 as the I/O access agent accepts an I/O request issued by the user, whose request is being processed by the main program as is described above in Figure 4. As the I/O access agent accepts the I/O access request at step 510, the program flow then proceeds to step 512 so as to register the very access request in the I/O process queue.

Then, step 514 checks to see if the I/O process queue maintained by the system is empty. A negative test result of step 514 indicates that there is at least one or more pending I/O access requests still waiting to be processed, and the program flow proceeds to step 516 in order to decide the type of the pending I/O access request. Then, in step 520, the I/O request is discriminated into either a read or a write request. For a read request, the program flow proceeds to step 530. For a write, the program transfers to step 522.

In case of a read-type I/O request, the program flow checks at step 530 to see if the information to be read-requested has already been present in the cache area of the storage subsystem. A recent previous read access to the same piece of information leaves a cache record in the storage subsystem. Such caches improve system overall performance effectively. If a cache record is indeed in existence, the program flow can be transferred to step 550, where the accessed information can be readily relayed back to the requesting user directly from the cache, and the program flow can be concluded for this particular read I/O request.

On the other hand, if the read-requested information is, unfortunately, not cached, the program flow proceeds to step 532, where the requested information is actually

located in the storage subsystem and needs to be retrieved. The program then waits for the retrieval of the target information to be concluded at step 534. Once concluded, program flow goes to step 540 and updates the read cache of the storage subsystem. Then, the read-type I/O request can be concluded at step 550 as described above.

5 In case of a write-type I/O request, the program flow proceeds from step 520 to 522, where the information provided by the user and asked to be written into the storage subsystem of the information service system is actually written. Then, as the program flow concludes the writing at step 524, the program flow can then be advanced to step 540, where the cache of the storage subsystem can be updated. At this moment, the write-type I/O access request is concluded, and the program flow can then be transferred to step 550 to conclude the processing as described above. Note here that once the program flow of Figure 5 is concluded at step 550, the system control can be transferred from the I/O access agent back to the main program.

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995
1000
1005
1010
1015
1020
1025
1030
1035
1040
1045
1050
1055
1060
1065
1070
1075
1080
1085
1090
1095
1100
1105
1110
1115
1120
1125
1130
1135
1140
1145
1150
1155
1160
1165
1170
1175
1180
1185
1190
1195
1200
1205
1210
1215
1220
1225
1230
1235
1240
1245
1250
1255
1260
1265
1270
1275
1280
1285
1290
1295
1300
1305
1310
1315
1320
1325
1330
1335
1340
1345
1350
1355
1360
1365
1370
1375
1380
1385
1390
1395
1400
1405
1410
1415
1420
1425
1430
1435
1440
1445
1450
1455
1460
1465
1470
1475
1480
1485
1490
1495
1500
1505
1510
1515
1520
1525
1530
1535
1540
1545
1550
1555
1560
1565
1570
1575
1580
1585
1590
1595
1600
1605
1610
1615
1620
1625
1630
1635
1640
1645
1650
1655
1660
1665
1670
1675
1680
1685
1690
1695
1700
1705
1710
1715
1720
1725
1730
1735
1740
1745
1750
1755
1760
1765
1770
1775
1780
1785
1790
1795
1800
1805
1810
1815
1820
1825
1830
1835
1840
1845
1850
1855
1860
1865
1870
1875
1880
1885
1890
1895
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050
2055
2060
2065
2070
2075
2080
2085
2090
2095
2100
2105
2110
2115
2120
2125
2130
2135
2140
2145
2150
2155
2160
2165
2170
2175
2180
2185
2190
2195
2200
2205
2210
2215
2220
2225
2230
2235
2240
2245
2250
2255
2260
2265
2270
2275
2280
2285
2290
2295
2300
2305
2310
2315
2320
2325
2330
2335
2340
2345
2350
2355
2360
2365
2370
2375
2380
2385
2390
2395
2400
2405
2410
2415
2420
2425
2430
2435
2440
2445
2450
2455
2460
2465
2470
2475
2480
2485
2490
2495
2500
2505
2510
2515
2520
2525
2530
2535
2540
2545
2550
2555
2560
2565
2570
2575
2580
2585
2590
2595
2600
2605
2610
2615
2620
2625
2630
2635
2640
2645
2650
2655
2660
2665
2670
2675
2680
2685
2690
2695
2700
2705
2710
2715
2720
2725
2730
2735
2740
2745
2750
2755
2760
2765
2770
2775
2780
2785
2790
2795
2800
2805
2810
2815
2820
2825
2830
2835
2840
2845
2850
2855
2860
2865
2870
2875
2880
2885
2890
2895
2900
2905
2910
2915
2920
2925
2930
2935
2940
2945
2950
2955
2960
2965
2970
2975
2980
2985
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9490
9495
9500
9505
9510
9515
9520
9525
9530
9535
9540
9545
9550
9555
9560
9565
9570
9575
9580
9585
9590
9595
9600
9605
9610
9615
9620
9625
9630
9635
9640
9645
9650
9655
9660
9665
9670
9675
9680
9685
9690
9695
9700
9705
9710
9715
9720
9725
9730
9735
9740
9745
9750
9755
9760
9765
9770
9775
9780
9785
9790
9795
9800
9805
9810
9815
9820
9825
9830
9835
9840
9845
9850
9855
9860
9865
9870
9875
9880
9885
9890
9895
9900
9905
9910
9915
9920
9925
9930
9935
9940
9945
9950
9955
9960
9965
9970
9975
9980
9985
9990
9995
10000
10005
10010
10015
10020
10025
10030
10035
10040
10045
10050
10055
10060
10065
10070
10075
10080
10085
10090
10095
10100
10105
10110
10115
10120
10125
10130
10135
10140
10145
10150
10155
10160
10165
10170
10175
10180
10185
10190
10195
10

of different types, an I/O access agent subsystem can be set up between a main program subsystem and a storage subsystem having a number of organized storage devices. Parallel service processing can thus be achieved in a large information service system of the present invention.

Figure 6 is a simplified block diagram schematically illustrating the configuration of an information service system 600 comprising an expanded I/O access agent subsystem 630 in accordance with a preferred embodiment of the present invention. The access agent subsystem 630 comprising of one or more I/O access agents is set up between the main program subsystem 620 and the storage subsystem 640 of the service system 600. The main program subsystem 620 may also be a construction made up of a number of main program modules 621, 622, ... 629. The storage subsystem 640, on the other hand, may be one with a number of storage devices 641, 642, ... and 649. Such are main program and storage subsystems suitable for providing expanded service capability.

As an example, consider the situation in which the service system 600 of Figure 6 is utilized in an on-line game service. As mentioned above, the multiple number of I/O access agents 731, 732, ... and 739 of the I/O access agent subsystem 730 of the system 700 of Figure 7 can be divided among different types of I/O access activities organized according to a pre-determined rule. For example, the I/O access agent 731 in the subsystem 730 can be assigned to be responsible for handling I/O accesses relating to user identification information. Likewise, agent 732 can be responsible for participating game players' information accesses categorized in accordance with geological territories in the virtual game world. Further, agent 739 can be responsible for user accesses with regard to user game status.

In the described embodiment of Figure 7, each of the main program subsystem 720 and the storage subsystem 740 can be constructed to a structural configuration similar to that of the I/O access agent subsystem 730. For example, the main program module 721 in subsystem 720 can be set up to process users' I/O access requests in relation to user ID information. Main program module 721 may submit its I/O access requests received from game players to the corresponding I/O access agent 731 in subsystem 730. Similarly, a dedicated storage device 741 in the storage subsystem 740

can be assigned for the storage of user ID-related information for all registered game players of the service.

Note that the block diagram of Figure 7 shows one single-block 741 for the storage of user ID information. However, as is comprehensible, a storage device 741 in subsystem 740 may be one single disk drive subsystem, or it may be a cluster of several server machines connected to a local area network if the user database is huge to be handled by a single server. Nonetheless, all storage devices in the subsystem 740 are integrated as a whole. This is much like how all the I/O access agents in the subsystem 730 and all the main program modules in the subsystem 720 are respectively integrated and organized for concerted operation in the service system 700 for hosting an on-line game service business.

Thus, the system and method of interruption-free file access as disclosed by this invention can be summarized to the principle of "let those have to wait keep waiting and allow those not need to wait keep going unaffected," which is a concept of non-blocking I/O. The system and method of the present invention are also suitable for expansion along with growing service requirements employing measures such as categorized and dispersed I/O processing among a large array of storage devices. Service system overall operating efficiency is therefore optimized.

In the prior art, a multiple-thread operating system is likely to be dragged into sluggishness whenever certain I/O-intensive processing thread is submitted. Such I/O-intensive threads inevitably compromise the performance of other non-I/O-intensive threads processed in the OS. Other threads under the OS requiring I/O service may sometimes be stalled completely.

However, if the system and method of the present invention are applied in a multiple-thread OS, those I/O-intensive threads will be deprived of their CPU attention temporarily. In other words, an I/O-intensive thread is forced to relinquish its CPU control temporarily so as to pass CPU control over to other threads under the OS. An I/O-intensive thread surrendered its CPU control temporarily will only be re-assigned CPU control when its I/O access result is ready. Whenever after the I/O access result is ready, the thread may continue and conclude. During the period in which the I/O-intensive thread waits for its results, the OS is not dragged slow since the CPU is

free to service all other threads under the OS.

By contrast, a prior art single-thread OS will be stalled completely whenever an I/O-intensive service is undertaken by the OS. Software routines such as one that services a multiple number of software clients will be stalled until an on-going I/O service is concluded. During this period, OS service to all other clients is virtually stopped.

If, however, the system and method the present invention are applied in a single-thread OS, this stalling can be avoided. The software system employing the inventive method receives an I/O-intensive service request, puts it into a control list and engages processing for the request, and then go on to service other software requirements instead of waiting for the time-consuming I/O service to conclude. This frees up the CPU from the tying-down of idling and waiting for implementing services to other requirements in the system. The only one put to wait is the I/O-intensive service request itself.

While the above is a full description of the specific embodiments, various modifications, alternative constructions and equivalents may be used. Therefore, the above description and illustrations should not be taken as limiting the scope of the present invention which is defined by the appended claims.